Conflict Resolution in Warehouse Robotics

Goutham Muralikrishnan
Department CSAI
University of Technology Nuremberg
Nuremberg, Germany
goutham.muralikrishnan@utn.de

Noah Steinkrüger
Department CSAI
University of Technology Nuremberg
Nuremberg, Germany
noah.steinkrueger@utn.de

Vinay Orsu
Department CSAI
University of Technology Nuremberg
Nuremberg, Germany
orsu.vinay@utn.de

***Abstract*—Mobile robots deployed in warehouses must navigate safely and efficiently in close proximity, sometimes without reliable global coordination. We present a fully decentralized conflict detection and resolution module that runs onboard each robot. During nominal navigation, each robot monitors LiDAR scans, odometry, commanded velocities. When a conflict is triggered, a lightweight quadratic program filters the nominal velocity using LiDAR-derived barrier constraints for collision avoidance and a liveness (yielding) constraint to break symmetric deadlocks. We evaluate the approach in multi-robot simulation scenarios with static and dynamic obstacles and randomized initialization perturbations. Results show our approach with the detection and resolution performed better when compared to a baseline without detection-controlled arbitration.**

***Index Terms*—multi-robot, conflict, resolution, detection, optimization**

I. Introduction

In real-world environments such as warehouses and offices, mobile robots must navigate under operational constraints. In these settings, robot interactions are not always explicitly coordinated, and navigation becomes difficult when obstacles are dynamic and the environment is only partially mapped. Many existing systems rely on rule-based heuristic decision-making when acting under uncertainty, which often leads to multi-robot conflicts and makes reliable navigation challenging.

Many existing approaches or state-of-the-art multi-robot navigation systems rely on a central planner/controller that performs path planning. It requires the global state [1] to navigate the robots without getting into conflicts. Although this simplifies coordination, it requires reliable communication and can become computationally expensive as the number of robots or the density of interactions increases [2]. Centralization introduces a single point of failure, where coordination and navigation performance can degrade significantly, if the server or network becomes unavailable.

More recently, foundational model-based approaches have been explored to improve decision-making [3], [4]. These methods do not automatically guarantee feasibility at the control level. Deploying such models in real-world navigation introduces new challenges such as high runtime-latency, compute constraints, and potential reliability issues. Some hybrid approaches mitigate this by running local policies and switching to a learned coordination strategy when a deadlock is detected [2].

Here we present a decentralized system where the conflict detection and resolution layer runs on-board each robot and integrates with the standard Nav2 navigation pipeline [5]. Each robot follows Nav2's nominal global plan. Our low-level controller continuously monitors safety and progress using LiDAR scans, odometry, commanded velocities, and plan-derived progress signals from Nav2. When a conflict is detected, our resolution module computes a safety-filtered velocity command by optimizing a quadratic function [6], [7], based on a lightweight control barrier. This enforces collision avoidance while preserving the nominal command as much as possible. Our approach avoids reliance on a centralized controller or coordination and remains effective in dense and dynamic navigation zones. We evaluate the proposed system in multi-robot simulation setups and compare it with the baseline.

II. Related Work

A. Conflict Detection

A common run-time detection declares a *conflict* when another robot or obstacle penetrates a predefined safety buffer around the robot. Ferrera *et al.* model each agent with an inflated circular safety zone and detect collisions via zone intersection, yielding a decentralized criterion that is tolerant to moderate pose uncertainty [8]. In the same approach, SWAP [9] uses local range-finder measurements to test whether the robot's safety region intersects an *inflated* obstacle representation, where inflation provides conservatism for braking distance and sensing uncertainty. SWAP further shows these interactions in polar sensor space and maps safety-region intersections to *forbidden angular sectors* (unsafe headings), directly coupling conflict detection with reactive velocity selection [9]. While full-around monitoring with conservative inflation improves safety, it can be overly restrictive in tight environments, motivating sectorized or motion-conditioned safety checks.

Detection is also detected at the planning layer, where conflicts are typically defined as *vertex* or *edge* collisions in space–time. Kollakidou and Bodenhagen extend Conflict-Based Search (CBS) style conflict checks from a single timestep to a *rolling time window*, so that small execution delays or speed variations are more likely to be flagged before deployment [10]. They further adapt the window size based on factors such as map structure and problem density, em-

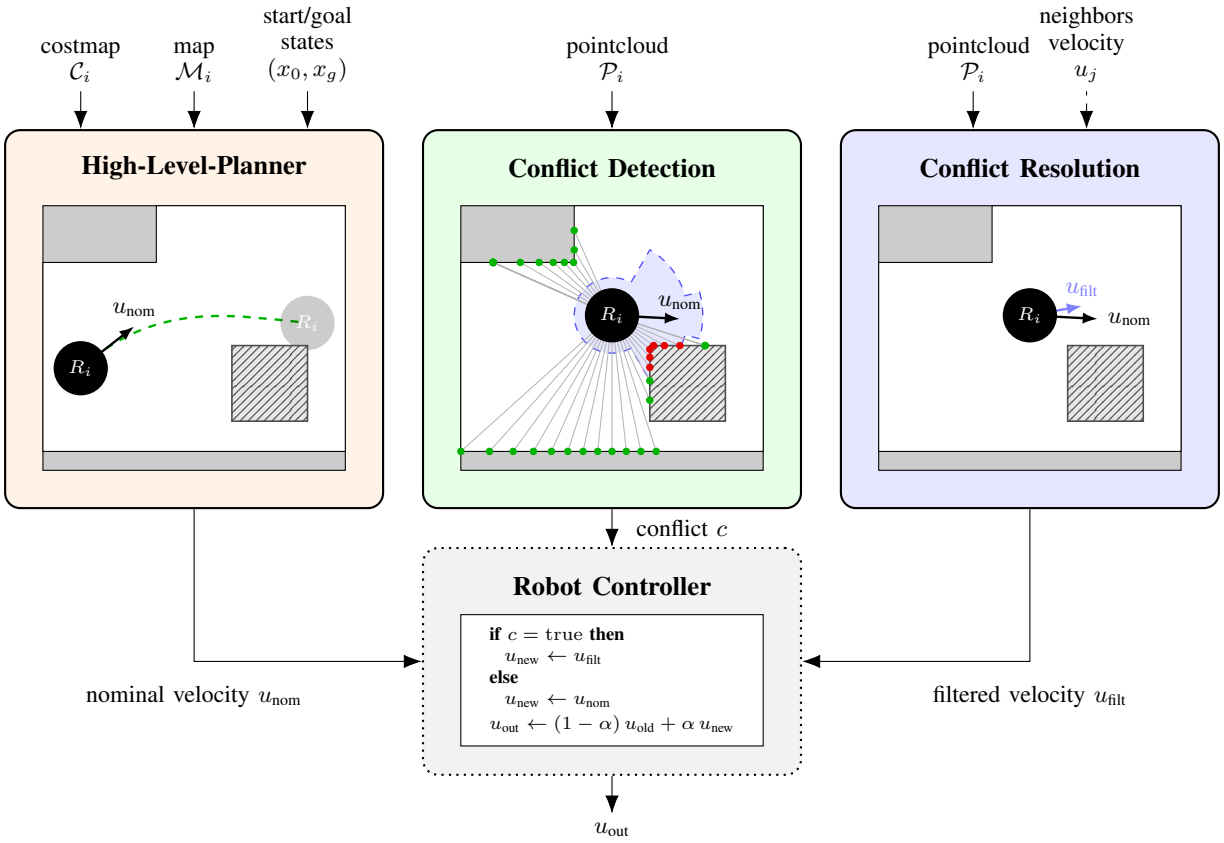

Fig. 1. System architecture of the velocity generation and arbitration pipeline. The *High-Level Planner* receives the local map $\mathcal{M}_i$, costmap $\mathcal{C}_i$, and start/goal states (x_0, x_g), and produces a nominal velocity command u_{nom}. In parallel, the *Conflict Detection* module processes the current LiDAR scan $\mathcal{S}_i$ to determine whether a collision risk exists, outputting conflict flag c. The *Conflict Resolution* module uses the same scan $\mathcal{S}_i$ together with optional neighboring agents' velocities u_j to compute a filtered, collision-avoiding velocity u_{filt}. The *Robot Controller* selects between u_{nom} and u_{filt} depending on the conflict flag c, and applies a smoothing update $u_{\text{out}} \leftarrow (1-\alpha)u_{\text{old}} + \alpha u_{\text{new}}$ to generate the final output command u_{out}, which is sent to the robot actuators.

phasizing that practical detectors often need explicit temporal slack.

Beyond imminent collision, several systems trigger higher-level intervention when robots cease making progress. Garg *et al.* define a deadlock when the team's average speed drops below a threshold while the average distance to goals remains above a minimum, and use this condition to activate a high-level planner [11]. Such progress-based criteria complement proximity triggers by capturing congestion and local minima that do not necessarily involve near-field contact.

Considering these approaches, we adopt safety-region reasoning as the primary cue for detection, but modify the monitored geometry to the forward region rather than treating the full perimeter uniformly.

B. Conflict Resolution

Centralized coordination approaches, including MAPF-style planners, resolve multi-robot conflicts by computing globally consistent paths under a shared world state [1]. While effective in structured settings, these methods typically assume reliable global information and can scale poorly in dense interactions, where the coupled planning space grows quickly, and coordination becomes sensitive to communication and server availability [2].

A widely used alternative is decentralized reactive collision avoidance, where each robot selects a collision-free velocity directly in the control space [12], [13]. These methods are attractive due to low runtime and distributed execution, but they can become overly conservative in narrow passages and may stall in highly symmetric encounters where agents must be effectively re-ordered rather than simply sidestepping [7]. In such cases, robots often slow down, oscillate, or freeze, and progress stops even without imminent collision [7].

To address deadlocks that purely reactive methods cannot solve, hybrid systems run a lightweight local policy under normal operation and switch to a stronger coordination module once deadlock is detected [2]. A representative strategy integrates a reactive stack (e.g., ORCA combined with a global planner) and triggers a local MAPF sub-solver (e.g., Push-and-Rotate or ECBS) only for the subset of robots involved in the conflict [1], [2], [13]. This can improve liveness in dense interactions while avoiding constant global joint planning. However, dense multi-robot scenarios increase integration complexity and can introduce unpredictable compute spikes at runtime, particularly when conflicts occur frequently or involve multiple robots [2].

Considering these trade-offs, our resolution module follows a lightweight decentralized design compatible with standard navigation stacks. Each robot uses its onboard LiDAR obser-

vations of neighboring robots to estimate their short-horizon motion and filter the commanded velocity accordingly.

III. Methods

We consider a team of differential-drive robots operating in a shared environment. Each robot R_i receives a map $\mathcal{M}_i$, start $x_0^{(i)}$, and goal pose $x_g^{(i)}$. A high-level planner provides a path and nominal velocity command:

$$u_{nom} = [v_{nom}, w_{nom}]^\top \tag{1}$$

We consider scenarios where the planned trajectories between the robots overlap or have obstacles on the path during the navigation to the goal as conflict scenario. Aim of our approach is to navigate each robot to its goal without collision or ending up in a deadlock. Fig. 1 illustrates the overall control pipeline.

A. Conflict Detection

Conflict detection determines when a robot can no longer follow the nominal navigation command safely or make meaningful progress toward its goal.

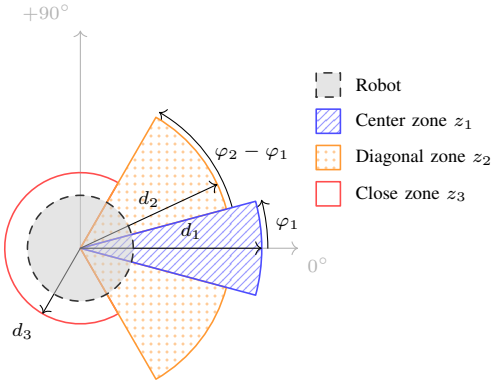

Fig. 2. Working principle of the conflict detection. The used safety zones z_i with $i \in \{1, 2, 3\}$ for the lidar data to flag a conflict are defined by their angle φ_i and their radius d_i

All detectors operate locally (no inter-robot communication) and use:

- forward LiDAR scans for near-field proximity,
- Nav2 global plan and remaining distance-to-goal,
- commanded velocities, and
- odometry/pose for progress verification.

1) Safety-Region Detector (LiDAR): As illustrated in Fig. 2, the safety-region detector monitors a forward LiDAR field and raises a conflict when obstacles intrude within conservative stopping distances. The detector is reactive and independent of the global plan, which makes it robust to planning uncertainty and communication outages. To reduce false positives during intentional in-place rotation (common during local replanning), the detector is gated by commanded motion and may be suppressed when the robot is commanded to rotate in place without forward motion.

a) Region decomposition: We define three LiDAR safety regions (Fig. 2): a forward *center* sector z_1, a forward *diagonal* sector z_2, and a 360° *close-proximity* zone z_3 for imminent hazards:

$$z_1 = \{\phi : |\phi| \le \varphi_1\}, \tag{2}$$
$$z_2 = \{\phi : \varphi_1 < |\phi| \le \varphi_2\}, \tag{3}$$
$$z_3 = \{\phi : -\pi < \phi \le \pi\}. \tag{4}$$

Let $r(\phi)$ denote the LiDAR range at angle ϕ. We define hit indicators:

$$\text{hit}_i(\phi) = \mathbb{I}\big(r(\phi) < d_i\big), \quad \phi \in z_i, \forall i \in \{1, 2, 3\} \tag{5}$$

b) Conflict flag: We distinguish between (i) a *conflict flag* that triggers the resolution module and (ii) an optional conservative stopping rule used during detection evaluation. Each $\text{hit}_i(\phi)$ has its own threshold which decides to flag as conflict if the number of hits for any of them crosses a threshold value. During detection evaluation, we used a *fully blocked* condition, when all the zones hits have crossed the threshold. This is to enforce an immediate stop for collision prevention. In the final integrated system with resolution, we trigger conflict resolution, and the resolution module computes a safe filtered velocity.

2) Mini-Goal Progress Detector (Plan-Based): To capture non-collision stalls (e.g., congestion or local minima) without near-field obstacles, we sample *mini-goals* along the global plan. The plan is partitioned into a small set of intermediate checkpoints $\{g_1, \ldots, g_M\}$ that the robot should reach sequentially. A mini-goal is considered reached when the robot enters a radius ϵ_g around the checkpoint. A conflict is raised if the robot fails to reach the current mini-goal within a time budget derived from the planned path length and an estimated traversal speed. This detector, therefore, depends on the availability and stability of the global plan and is primarily intended to detect stalled progress even when the near field is free.

3) Distance/Velocity Stagnation Detector: This detector flags conflicts when the robot is commanded to move but makes negligible progress toward the goal. Progress is measured using the remaining distance on the path toward the goal given by high-level planner. Wheel/odometry feedback is used to differentiate true stalls from cases where progress is slow but still occurring. To avoid transient triggers immediately after replanning, a short grace period can be applied after receiving a new global plan.

B. Conflict Resolution

The approach we follow aims to compute a filtered velocity

$$u_{\text{filt}} = [v, \omega]^\top \tag{6}$$

which guarantees collision avoidance and resolves multi-robot conflicts. We assume a differential drive kinematic model. We define our objective as minimizing the deviation of this filtered velocity u_{filt} and the nominal velocity u_{nom}

$$\min[(v - v_{\text{nom}})^2 + \lambda_\omega(\omega - \omega_{\text{nom}})^2] \tag{7}$$

where λ_ω is a parameter that allows to either encourage or suppress angular motion relative to linear motion. The constraints of our optimization problem guide the resolving behavior as well as collision avoidance.

1) Safety Constraints from LiDAR: At each control cycle, a robot processes its LiDAR scan to obtain a point cloud

$$\mathcal{P} = \{(x_k, y_k)\}_{k=1}^{M} \tag{8}$$

in the robot base frame. We filter those points by range and for each angular bin select the closest point. Out of this reduced set, we restrict the points further to the closest $M_{\max}$ points. For each point k, we define a distance-based barrier function

$$h_k(x_k, y_k) = x_k^2 + y_k^2 - \delta_{\text{eff}}^2 \tag{9}$$

where the effective safety margin

$$\delta_{\text{eff}} = \delta + \tau \max(0, v_{\text{nom}}) \tag{10}$$

considers the clearance proportional to the nominal forward speed. The parameter τ determines how much the forward nominal speed should be considered. Our barrier function h_k must satisfy the constraint

$$h_k \geq 0 \tag{11}$$

However, satisfying this constraint at the current timestep is not sufficient. We additionally must ensure that the robot's motion does not drive the robot into the unsafe region in the next timestep. We enforce this by adding a constraint to ensure forward invariance

$$\dot{h}_k + \alpha h_k \geq 0 \tag{12}$$

where α is a hyperparameter that determines how aggressively the system is pushed away from the boundary. If h_k is large, which means the robot is far from any obstacle, the constraint is loose. The constraint becomes strict in the case of lower values for h_k. With the approximation of $\dot{h}_k$

$$\dot{h}_k \approx -2x_k v - 2y_k \omega L \tag{13}$$

where L is a lookahead distance, we get the full contraint as

$$-2x_k v - 2y_k \omega L + s_{\text{safe}} \geq -\alpha h_k \tag{14}$$

We use this approximation to ensure real-time QP solving.

2) Deadlock Detection and Liveness Modeling: Each time a robot R_i detects a conflict it further filters the conflict and identifies a dominant neighbor R_j within a deadlock radius. A deadlock is further detected using two complementary criteria:

- Geometric liveness condition which is based on the relative position $\mathbf{r}_{ij}$ and the relative velocity $\mathbf{v}_{ij}$ (Fig. 3)

$$\theta_{ij} = \arccos\left(\frac{\mathbf{r}_{ij}^\top \mathbf{v}_{ij}}{\| \mathbf{r}_{ij} \| \| \mathbf{v}_{ij} \| + \epsilon}\right) \tag{15}$$

  For small liveness angles θ_{ij} we detect a deadlock.
- Or a progress-based condition triggers the deadlock if

$$|v| < v_{\min} \quad \text{and} \quad |v_{\text{nom}}| > v_{\text{try}} \tag{16}$$

  holds for longer than a prescribed time window.

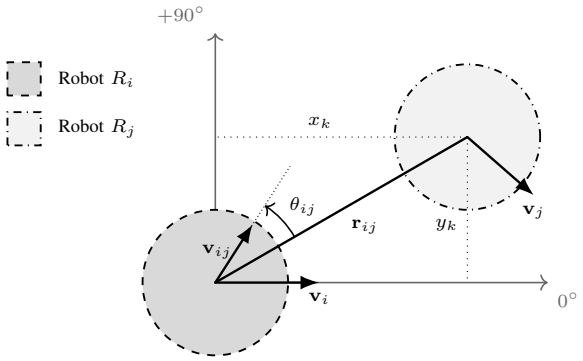

Fig. 3. Robot R_i is located at the origin with velocity $\mathbf{v}_i$, while a neighboring robot R_j is positioned relatively at $\mathbf{r}_{ij}$ and moves with velocity $\mathbf{v}_j$. The relative velocity $\mathbf{v}_{ij}$ describes the motion of robot R_j with respect to robot R_i. The liveness angle θ_{ij} is defined between the relative position vector $\mathbf{r}_{ij}$ and the relative velocity vector $\mathbf{v}_{ij}$, indicating whether the robots are moving toward or away from each other. The dotted projections illustrate the decomposition of the neighbor's position into local coordinate components (x_k, y_k).

3) Yielding Constraint: To resolve the deadlock scenarios each robot decides deterministically whether to yield or proceed. For example based on priority or robot id if communication is allowed, else the robot always yield to its dominant neighbor j. Without communication the dominant neighbor is determined geometrically and its velocity is estimated by LiDAR tracking. The forward velocity of the robot i is constrained as

$$v \leq \frac{1}{\zeta} \| v_j \| + s_{\text{live}} \tag{17}$$

where $s_{\text{live}} \geq 0$ is a liveness slack variable and $\zeta > 1$ a hyperparameter that enforces strict speed reduction. This guarantees that at least one robot maintains forward progress.

4) Quadratic Program Formulation: Now we combine the main objective (7) with slack variables enforcing safety s_{safe} and liveness s_{live} constraints:

$$\mathcal{S} = (v - v_{\text{nom}})^2 + \lambda_\omega(\omega - \omega_{\text{nom}})^2 + \rho_{\text{safe}} s_{\text{safe}}^2 + \rho_{\text{live}} s_{\text{live}}^2 \tag{18}$$

The related parameters ρ_{safe} and ρ_{live} penalize the constraints. ρ_{live} is chosen large enough to enforce yielding when required, but typically smaller than ρ_{safe} to ensure that collision avoidance dominates deadlock resolution. In the quadratic program we combine the objective (18) and the constraints (14), (17) introduced before to

$$\begin{aligned}
\min_{v, w, s_{\text{safe}}, s_{\text{live}}} \quad & \mathcal{S}(v, w, s_{\text{safe}}, s_{\text{live}}) \\
\text{s.t.} \quad & -2x_k v - 2y_k \omega L + s_{\text{safe}} \geq -\alpha h_k, && \forall k \in \mathcal{P} \\
& v \leq 1/\zeta \cdot \| v_j \| + s_{\text{live}}, && \text{if yielding} \\
& v_{\min} \leq v \leq v_{\max},\ \omega_{\min} \leq \omega \leq \omega_{\max}, \\
& s_{\text{safe}}, s_{\text{live}} \geq 0
\end{aligned}$$

5) Robustness Enhancements: In conflicts with missing neighbors or scenarios in which communication between robots is not possible, neighbor states are predicted using a constant-velocity model and missing neighbors are approximated using LiDAR-based clustering to form proxy agents. Additionally, when progress stalls at close range, a short-horizon escape maneuver temporarily overrides the nominal command to rotate, after which the optimization from earlier resumes.

C. Robot Controller

The robot receives a signal indicating whether it is in a conflict from the conflict detection module. If a conflict is detected, the resolution module computes a filtered velocity based on the nominal velocity command provided by the high-level planner. To prevent jittering or oscillations between the conflict and non-conflict states, we introduced a smoothing filter. Specifically, the output velocity is computed α-weighted update of the previous command velocity and the newly computed command velocity.

IV. Experiments

To evaluate our approach, we designed a set of scenarios with increasing complexity in terms of the number of robots, the environment topology, and the presence of static or dynamic obstacles. In total, we tested four different environments: a corridor, a crossing, a T-junction, and a warehouse with different lanes and obstacles. The corridor environment isolates head-on conflicts and obstacle avoidance in constrained spaces. The crossing environment models a four-way intersection, introducing multi-directional conflicts. The T-junction features an asymmetric topology and provides additional space, allowing experiments with a higher number of robots. We first evaluated our detection approach without any resolution and then combined both to evaluate them together. All hyperparameters used were tuned manually and remain constant for all scenarios.

A. Detection Evaluation

For each scenario type, we executed four runs: one nominal run with no spawn offset, and three runs with randomized spawn offsets (± 0.3–$0.5\,\mathrm{m}$ forward/lateral) and yaw perturbations ($\pm 30^\circ$) to evaluate robustness to initialization uncertainty.

A run is counted as *Success* if robots reach the intended conflict zone and a conflict is detected in time to prevent collision or unsafe behavior. A run is counted as *Failure* if a collision occurs without timely detection. Runs are marked *N/A* when the scenario does not complete or no valid conflict event occurs (e.g., conservative early stopping prevents reaching the intended conflict).

Table I summarizes detection outcomes across scenarios with multiple robots and obstacles. Across all nominal (no-offset) runs, robots consistently reached the intended conflict zone and conflicts were detected by V1 (LiDAR safety region). The LiDAR safety-region detector reliably triggered prior to impact, while plan-dependent progress detectors typically triggered only after contact (once motion ceased and progress metrics degraded). When offsets were introduced, results became more variable due to high-level path uncertainty: in some runs robots planned closer to walls or static structure, causing early LiDAR triggers that prevented reaching the intended conflict zone (counted as N/A). Some cases led to failures when the detection did not prevent from collision.

TABLE I
Conflict detection results across scenarios (simulation). Each scenario was evaluated over 4 runs.

Scenario	R	O	Det.	N/A	S	F	SR (↑)
Two Lane	2	1	V1	3	1	0	1.00
Two Lane	3	1	V1	3	1	0	1.00
Two Lane	3	2	V1	2	2	0	1.00
Single Lane	1	1	V1	3	1	0	1.00
Single Lane	2	1	V1	1	1	2	0.33
Three Lane	3	1	V1	0	2	2	0.50
Three Lane	3	2	V1	1	3	0	1.00
Three Lane	3	3	V1	0	3	1	0.75
Intersection	2	0	V1	3	1	0	1.00
Intersection	3	1	V1	2	2	0	1.00
Intersection	4	0	V1	2	2	0	1.00
Intersection	4	1	V1	0	4	0	1.00

Abbreviations: R = Robots, O = Obstacles, Det. = Detector Triggered (V1/V2/V3), N/A = No valid conflict occurred, S = Successes, F = Failed, SR = Success Rate.

As the LiDAR safety-region detector (V1) consistently provided the earliest and most reliable indication of imminent collision. Consequently, we refined the forward safety sector and added the near-field close zone z_3 after the detection evaluation, and used LiDAR-based safety region V1 detection as the primary conflict trigger during conflict resolution.

B. Resolution Evaluation

For each of the predefined conflict scenarios described above, we conducted 10 runs. We recorded collisions, whether each robot reached its goal, and every instance in which a robot flagged or unflagged a conflict, which can occur multiple times within a single scenario. The resolution time T_{res} for an experiment is defined as the interval between the first timestep at which a robot detects a conflict and the last timestep at which any robot flags the conflict as solved

$$T_{\text{res}} = \max_i t_{\text{end}}^{(i)} \; - \; \min_i t_{\text{start}}^{(i)}. \tag{19}$$

provided the scenario was successfully resolved. Success is defined as all robots reaching their goals within a specified timeframe and without collisions. Table II lists these metrics for all tested scenarios. The experiments are categorized into three groups: scenarios with no obstacles besides other robots, scenarios with static obstacles, and scenarios with dynamic obstacles. In the dynamic-obstacle scenarios, obstacles move continuously across the robot's path, which can repeatedly lead to collisions while the robot attempts to replan and escape. As a result, several trials reached the goal but incurred one or more collisions. In complex warehouse-like scenarios, the success rate decreased due to constrained free space

TABLE II
RESOLUTION RESULTS. EACH SCENARIO WAS EVALUATED OVER 10 RUNS.
"–" INDICATES UNDEFINED (NO SUCCESSFUL RUN TO COMPUTE T_{RES}) OR NOT EVALUATED FOR THAT CONFIGURATION.

	Environment	# of robots	# of obstacles	Detection+Resolution		Resolution	
				SR (↑)	T_{res} (↓) [s]	SR (↑)	T_{res} (↓) [s]
static	Corridor	2	0	1.00	20.35 ± 15.34	–	–
	Crossing	2	0	0.70	39.53 ± 29.50	0.50	0.49 ± 0.63
	Crossing	3	0	0.90	66.29 ± 55.16	0.10	6.13 ± 0.00
	Crossing	4	0	0.40	62.27 ± 41.47	0.0	–
	Corridor	1	1	1.00	17.27 ± 8.18	1.00	0.53 ± 0.69
	T-junction	3	1	0.80	52.13 ± 22.89	–	–
	T-junction	5	1	0.80	116.33 ± 31.20	0.0	–
	Warehouse	2	1	0.30	97.55 ± 79.11	0.0	–
	Warehouse	3	1	0.10	124.98 ± 0.0	–	–
dyn	Corridor	1	1	0.00	–	0.00	–
	Crossing	3	1	0.10	135.93 ± 0.00	–	–

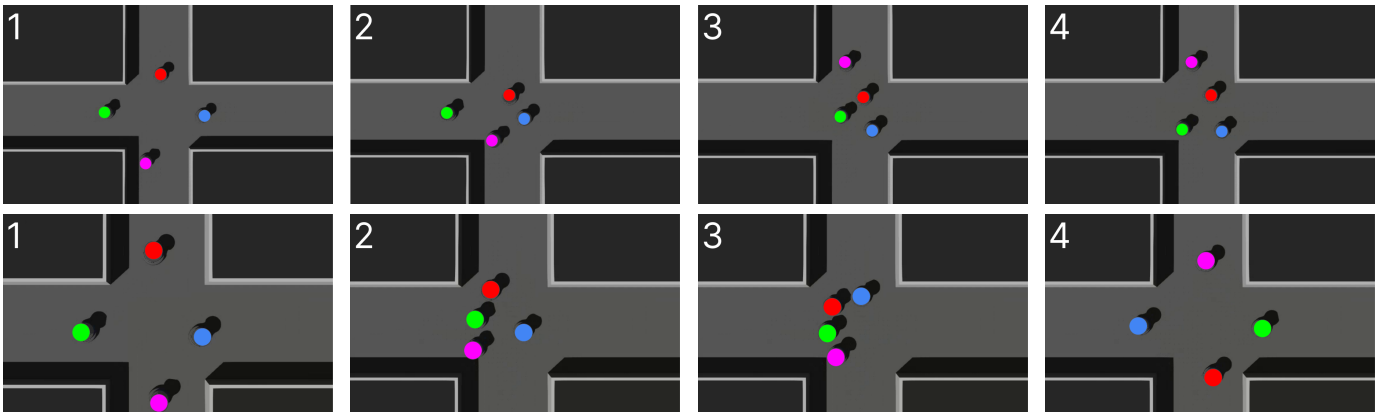

Fig. 4. Comparison of resolution-only (top row) and conflict-detection + resolution (bottom row) runs at successive stages. In the resolution-only sequence (top row, left-to-right) , the robots converge and remain stuck at the intersection; with conflict detection + resolution sequence (bottom row, left-to-right) , the robots are able to yield and reach their goals. Each robot is denoted by a different color. The goal of each robot is to reach the other side of the crossing.

and increased interaction density. Narrow passages reduce the set of viable collision-free moves for differential-drive robots, while concurrent robots create correlative blocking that triggers deadlocks and oscillatory re-planning without any progress. As a result, several runs have been terminated because of the timeout, which is also the reason for lower success rates in other scenarios.

C. Comparison of our method with baseline

We evaluated the baseline approach under identical experimental conditions. The baseline approach uses the conflict resolution module directly without conflict detection, controlling the switch between the high-level planner and filtered velocity. As shown in Table II and Fig. 4, the baseline exhibits significantly lower performance across most scenarios. In several cases, robots either remained stuck for extended periods or failed to reach the goal entirely. We evaluated the baseline using the same metrics as our method, including collision-free success rate and conflict resolution time T_{res}.

These results highlight the importance of adaptive switching between navigation and conflict detection, which leads to good performance in multi-robot and dynamic scenarios.

V. CONCLUSION

In this paper, we presented a fully decentralized two-layer approach for multi-robot conflict resolution, where each robot detects conflicts locally and resolves them by filtering its nominal filter command using LiDAR observations of neighboring robots. A key contribution is combining the detection with the resolution, enabling conflict-aware behavior without replacing the underlying planner. We evaluated the system in simulation across multiple scenarios and reported collision-free task completion and resolution time. The approach assumes reliable LiDAR visibility and may degrade under occlusions (e.g., robots/shelves blocking returns in narrow aisles) or sensor dropout. Dense layouts can also limit collision-free maneuvers, which leads to robots repeated switching between filtered and nominal velocity. Future work could focus on parameter tuning, extending evaluation to more complex and dynamic environments, and improving handling of challenging interaction patterns that require stronger resolution and transferring the approach to real robots.

REFERENCES

[1] B. de Wilde, A. ter Mors, and C. Witteveen, "Push and rotate: Cooperative multi-agent path planning," in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2013, pp. 87–94.

[2] H. Wang, L. Luo, Y. Kantaros, B. Sinopoli, and M. Cai, "Deadlock-free hybrid rl-mapf framework for zero-shot multi-robot navigation," 2025.

[3] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine, "Vint: A foundation model for visual navigation," 2023.

[4] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," 2023.

[5] S. Macenski, F. Martin, R. White, and J. Ginés Clavero, "The marathon 2: A navigation system," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[6] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[7] J. Chen and R. Chandra, "Livepoint: Fully decentralized, safe, deadlock-free multi-robot control in cluttered environments with high-dimensional inputs," 2025.

[8] E. Ferrera, Á. R. Castaño, J. Capitán, P. J. Marrón, and A. Ollero, "Multi-robot operation system with conflict resolution," in *ROBOT2013: First Iberian Robotics Conference*, ser. Advances in Intelligent Systems and Computing, M. A. Armada, A. Sanfeliu, and M. Ferre, Eds. Cham: Springer, 2014, vol. 252, pp. 407–419.

[9] E. Ferrera, J. Capitán, Á. R. Castaño, and P. J. Marrón, "Decentralized safe conflict resolution for multiple robots in dense scenarios," *Robotics and Autonomous Systems*, vol. 91, pp. 179–193, 2017.

[10] A. Kollakidou and L. Bodenhagen, "Adapted conflict detection for conflict-based search," in *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART)*, INSTICC. SciTePress, 2024, pp. 367–373.

[11] K. Garg, S. Zhang, J. Arkin, and C. Fan, "Foundation models to the rescue: Deadlock resolution in connected multi-robot systems," 2024. [Online]. Available: https://arxiv.org/abs/2404.06413

[12] J. van den Berg, M. C. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[13] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*. Springer, 2011, pp. 3–19.